Image Classification Applied to High Energy Physics Events

HONORS THESIS

Presented in Partial Fulfillment of the Requirements for Graduation with Honors Research Distinction in the Degree Bachelor of Science in the Undergraduate Colleges of The Ohio State University

By

Jonathan Timcheck

Undergraduate Program in Engineering Physics

The Ohio State University

2015

Thesis Committee:

Richard Hughes, Adviser

Brian Winer

Gregory Lafyatis

Abstract

The Large Hadron Collider (LHC) is being upgraded to produce proton-proton collisions at 13TeV at a higher luminosity by the end of 2015. While increasing the chances for rare, interesting phenomena, the new environment will be substantially nosier, making it much more difficult for traditional analyses, which rely on largely isolated particles, to extract the associated signals. Deep Convolutional Neural Networks (DCNNs), computational models inspired by the visual cortex, have greatly surpassed the performance of other methods in image classification competitions. This project aims to evaluate the usefulness of DCNNs in the new environment at the LHC by literally viewing events as images. This paper describes how detector information can be converted into images and how a DCNN can be trained and optimized to distinguish $t\bar{t}$ and $W + 4jets$, two well-understood types of events. Current results show a DCNN, with only calorimeter information, can achieve roughly equivalent performance to that of a traditional multivariate technique utilizing the full detector.

Acknowledgments

I express my deepest gratitude to my advisers and colleagues for their incredible support throughout my undergraduate education. I would also like to thank the Ohio Supercomputer Center for their computational resources and the Ohio State Engineering Experiment Station for their financial support.

Vita

2011.....North Allegheny Senior High School

2015.....B.S. Engineering Physics, The Ohio State University

2015 to present.....MASt Applied Mathematics, University of Cambridge

Publications

Timcheck, J. (2013). The Higgs Boson?. *The Journal of Undergraduate Research at Ohio State*, *3*.

Chatrchyan, S., Khachatryan, V., Sirunyan, A. M., Tumasyan, A., Adam, W., Bergauer, T., ... & Van Spilbeeck, A. (2013). Search for the standard model Higgs boson produced in association with a top-quark pair in pp collisions at the LHC. *Journal of High Energy Physics*, *2013*(5), 1-47.

Fields of Study

Major Field: Engineering Physics

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

In order to test theories of the fundamental particles and their interactions, the Large Hadron Collider (LHC) at CERN collides protons with high center-of-mass energy (Evans & Bryant, 2008). These events produce heavy, exotic particles which quickly decay before they may be directly observed. Two independent general-purpose detectors, the A Toroidal LHC Apparatus (ATLAS) and the Compact Muon Solenoid (CMS), record the properties of these decay products. Millions of archived events must then be carefully analyzed to search for rare phenomena of interest, which often appear very similar to more copiously produced background processes. To increase the chances of producing and observing rare phenomena, the LHC has been upgraded to produce 13TeV collisions at a higher luminosity by the end of 2015. However, this will also create a noisier environment in which classification is more difficult (Bruce et al., 2014).

Traditional Approach

Currently, analyses use hand-designed algorithms to identify the types and properties of particles observed in the detector: jet clustering algorithms, track finders, etc. (Chatrchyan et al., 2013). Engineered features, such as the energies of and angles between certain particles, are then computed and combined through some multivariate system to perform holistic classification of events. Artificial Neural Networks (ANNs),

powerful computational models inspired by the neural connectivity in the brain, are often chosen to perform this task for their high performance (Chatrchyan et al., 2013).

As a significant historic example, multivariate analysis techniques played an important role in providing the sensitivity necessary for the discovery of a particle consistent with the Standard Model Higgs boson in 2012 (Chatrchyan et al., 2012). The Higgs boson is associated with a mechanism that imbues the other fundamental particles with their masses and completes the Standard Model of particle physics, our best theory for the fundamental particles and their interactions thus far (Higgs, 1964). Analyses such as the search for top quarks produced in association with the Higgs boson ($t\bar{t}H$) now seek to measure specific properties of the Higgs boson; however, they have only been able to set limits on these quantities using ANNs (Chatrchyan et al., 2013). In order to push further, there is a need for yet more powerful analysis techniques. In addition, these new techniques must perform well in the nosier LHC environment.

Advances in Machine Learning

Recently, Deep Artificial Neural Networks (DANNs), ANNs consisting of many layers of neurons, have entered the realm of computational feasibility with new computing hardware (Krizhevsky, Sutskever, & Hinton, 2012). Deep Convolutional Neural Networks (DCNNs), a variant of DANNs inspired by the visual cortex, present an entirely different approach to the problem of distinguishing signal from background. Unlike ordinary ANNs used in high energy physics to combine the power of several engineered discriminating features, DCNNs are capable of learning compact hierarchical representations. DCNNs discover the important features in images on their own and

subsequently aggregate these features to perform classification (LeCun & Bengio, 1995). DCNNs have greatly surpassed the performance of ANN feature engineering approaches and other machine learning techniques in image classification competitions such as ImageNet (Krizhevsky, Sutskever, & Hinton, 2012). The goal of this project is to evaluate the effectiveness of DCNNs in the new challenging environment at the LHC. DCNNs may discover and exploit subtle features that have not yet been engineered, and they may handle the noisier environment more elegantly than current algorithms that rely on isolated objects.

Chapter 2: Deep Convolutional Neural Networks

A Deep Convolutional Neural Network (DCNN) is a supervised learning machine. It takes a two-dimensional image as input, performs some computations, and produces output values. A DCNN can be configured to output real values for regression tasks or values bounded between 0 and 1 for classification tasks: estimated probabilities for each class to which an image might belong. A DCNN learns to perform this task effectively by repeatedly iterating through a set of labeled training images and adjusting its parameters to reduce classification error.

A DCNN consists of several layers of artificial "neurons." An example DCNN is shown as Figure 1. Each neuron has weighted connections to some set of neurons in the previous layer, and each neuron produces an "activation" value that is available to neurons in the following layer. A neuron computes its activation value by putting the weighted sum of its input neurons' activations through a non-linear function, such as a sigmoid or hyperbolic tangent. In this way, a DCNN operates as a feed-forward network. The first layer of neuron activations in a DCNN are set to the pixel intensity values of an input image; the second layer of neurons computes their activations based on the first layer activations; the third layer of neurons computes their activations based on the second layer activations, and so on. The last layer of neuron activations is the output of the DCNN.

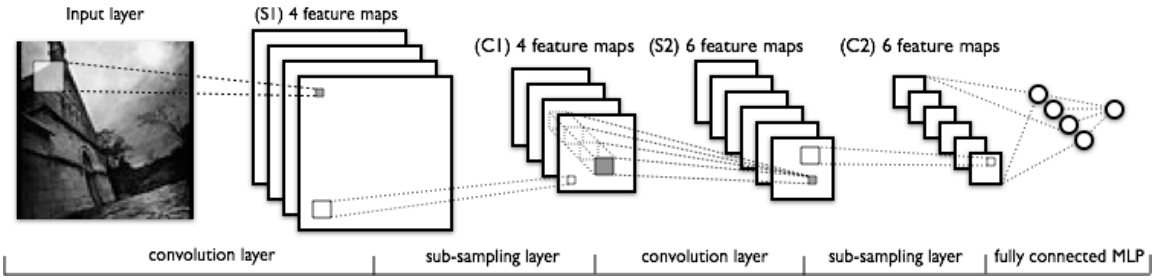

Figure 1: Example DCNN used to classify images ("Index of /tutorial/_images," 2015).

Connection weights are typically initialized in a random fashion, e.g., according to a Gaussian, to break the symmetry among the neurons (Krizhevsky, Sutskever, & Hinton, 2012). When put to the task of classification, a DCNN using randomized weights would perform comparably to guessing. In order to reduce its classification error, a DCNN adjusts its neuron connection weights. For classification tasks, the final neuron layer is given special Softmax activation functions, where each neuron represents a class assignment probability between 0 and 1, and the probabilities sum to 1 (Bridle, 1990). The Softmax error function can then be used to quantify how well the DCNN predicts the class of a given training image. More importantly, the partial derivative of the error with respect to a change in each weight in the network can be analytically expressed. The backpropagation algorithm can then be used to compute these derivatives efficiently (Lecun, Bottou, Orr, & Müller, 2012). Aggregating the derivates into a gradient, the error can be reduced to first order by taking a step opposite the gradient in the space of neuron connection weights as expressed in the following equation.

$$\vec{W} \to \vec{W} - \eta \vec{\nabla} E(\vec{W}) \quad (1)$$

$\vec{W}$ is the vector of connective weights in the network, $E(\vec{W})$ is the Softmax error, and η is the learning rate (an arbitrary parameter). The gradient may be computed for individual

input images and the weight update step carried out for each image, or the gradient may be averaged over several input images and the weight update step carried out on a batch-by-batch basis. Gradient descent does not guarantee convergence to a global minimum, but stochastic gradient descent, computing the gradient and updating on individual input images in a random order, has been shown to find good local minima in practice (Lecun, Bottou, Orr, & Müller, 2012).

The description of DCNNs thus far applies to any feed-forward Artificial Neural Network. The discriminating power of a DCNN arises due to the special configuration of its neurons. An input image is represented as a matrix of pixel intensity values, e.g., 0 for black, 255 for white, and values 1-254 for shades of gray, or a few matrices, one for each channel of color (typically red, blue, and green), and this constitutes the first layer of the DCNN. The second layer consists of neurons that convolve square receptive fields, e.g., 5 pixels by 5 pixels, over the image with strides of at least 1 pixel. These neurons compute an activation at each site and produce so-called feature maps, one map for each convolving neuron. The third layer is a subsampling layer in which the dimensionality of each feature map is reduced by periodically applying a simple aggregatory square kernel, such as maximum or average. After this layer, additional pairs of convolutional and subsampling layers may follow. All the neurons in the final subsampling layer are then fully connected to a layer of ordinary neurons. Any number of layers of fully connected neurons may follow. The last layer consists of the output neurons, whose activations correspond to estimated class membership probabilities in our case. (The fully connected

layers of neurons can be referred to as a Multilayer Perceptron (MLP) as labeled in Figure 1.)

In DCNNs, the consecutive convolutional/subsampling layers learn and extract hierarchal features in the input images, and the fully connected layers perform classification with these features. This configuration has many advantages over traditional neural networks simply consisting of several fully connected layers of neurons (ANNs) in the realm of image classification. The small collection of weights associated with each feature (E.g., for a 5x5 feature there are 25 weights.) is used repeatedly to produce an entire feature map. This is a substantial reduction in the number of parameters necessary to learn when compared to an analogous fully connected layer, and this reduction is valid because features are typically translationally invariant in images. In addition, the general principle of depth (number of layers) in artificial neural networks has shown the capability to express complex functions more efficiently (less neurons) than equivalent shallow networks (Bengio, Lamblin, Popovici, & Larochelle, 2007). This combination of attributes allows DCNNs to classify images very effectively.

Translating a HEP Detector into an Image

In order to apply DCNNs to the search for rare phenomena in high energy physics (HEP) detectors, we must convert detector information into an image. Typical HEP detectors like CMS can be thought of as containing two cylindrical grids of calorimeter towers that record the energy deposited by electromagnetically and hadronically interacting particles, respectively. Figure 2 shows a schematic of CMS. We unroll these grids and view them, overlaid, as a 2-dimensional, 2-channel intensity map. This is

equivalent to a flat 2-color image, which may be fed into a DCNN for classification. Our calorimeter grid images have dimensions of η, the pseudorapdity, and ϕ, the azimuthal angle. η and ϕ parametrize the space appropriately for highly boosted objects; in principle, the features in these images are translationally invariant as required for DCNNs. The calorimeters provide the most direct analogue to images, and including information from other subdetectors is described in Chapter 6.

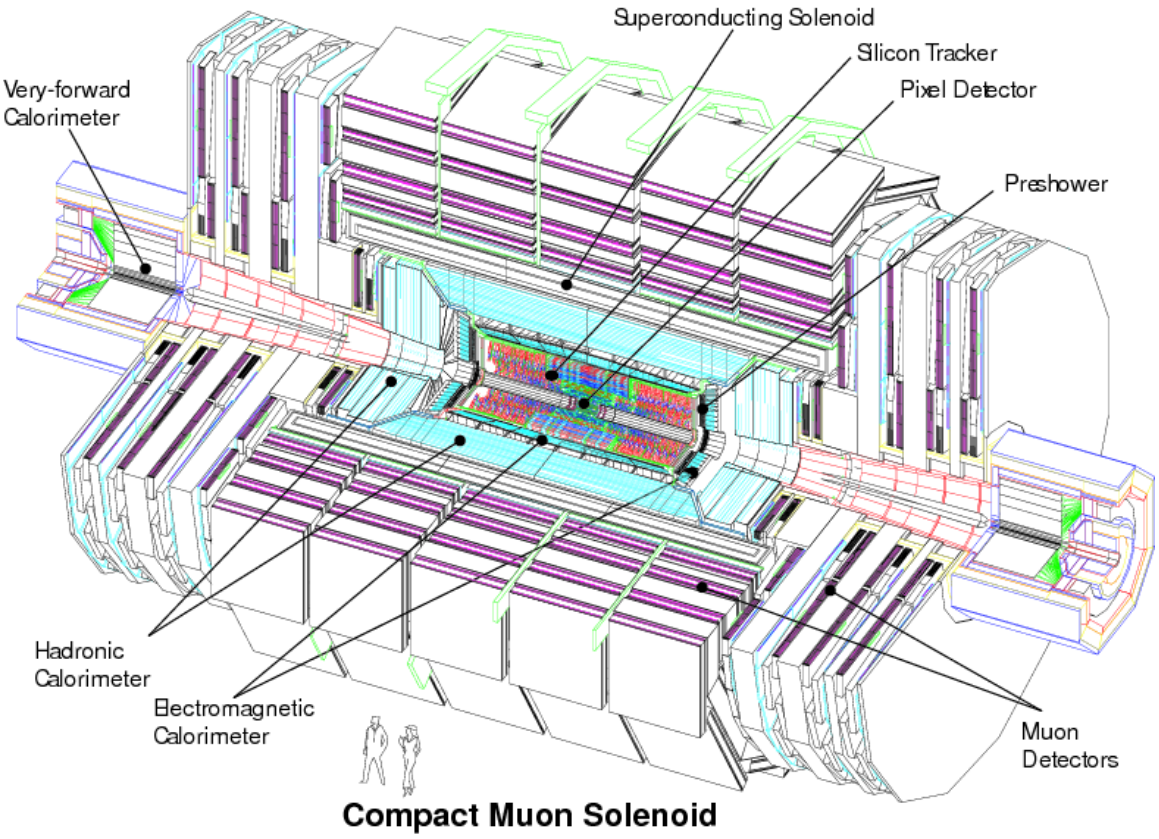

Figure 2: Schematic of CMS (Orimoto, 2009).

Chapter 3: Test Samples

To create a testing ground for our method, we use simulated samples of $t\bar{t}$ (signal) and $W + 4jets$ (background) at 13TeV. Studying the top quark, the heaviest fundamental particle, is a natural place to investigate the hierarchy problem and search for physics beyond the Standard Model (Bernreuther, 2008). Many top quark analyses have confronted the problem of distinguishing $t\bar{t}$ from the primary background process $W + 4jets$, and therefore much effort has been expended to ensure the accurate simulation of these processes (Aad et al., 2012). This setting provides an ideal context in which to test the idea of applying DCNNs to calorimeter images.

Two generations of $t\bar{t}$ and $W + 4jets$ samples have been studied throughout the duration of the project. The first generation consisted of 80,000 $t\bar{t}$ and 80,000 $W + 4jets$ events at 13TeV using the event generator Pythia 8 and the detector simulation package Delphes 3. Pythia 8 is a standard tool that generates high energy collisions according to theory (Sjöstrand, Mrenna, & Skands, 2008). Delphes 3 models a simplified version of ATLAS/CMS (de Favereau et al., 2013). The basic calorimeter in this virtual detector allowed us to directly fill floating point value pixels in a 2-channel 96x90 grid (96 η bins by 90 ϕ bins) with the energy deposited in the electromagnetic and hadronic calorimeters. We chose $-3 \leq \eta \leq 3$ and $0 \leq \phi \leq 2\pi$. The 96x90 images were extended

to 96x96 by repeating the first 6 ϕ rows at the bottom of the image so that feature wrap-around could occur.

The second generation consisted of 60,000 $t\bar{t}$ and 60,000 $W + 4jets$ samples at 13 TeV simulated in the CMS detector with pileup equal to 20 and 25 nanosecond bunch spacing, conditions that could be expected at the LHC by the end of 2015. These samples provided a more rigorous testing ground for our method. Events were generated using MadGraph, Pythia, TAUOLA, and TUNE4C. MadGraph generated the hard scatter: the 4-vectors of the top quarks, the W bosons, etc. (Alwall, Herquet, Maltoni, Mattelaer, & Stelzer, 2011). Pythia performed the fragmentation and decay of these particles, save for the tau leptons which TAUOLA handled. Pythia fine-tuned the underlying event with TUNE4C. We required that the simulated events pass minimum quality criteria: at least 4 jets with $Et > 25GeV$ (energy in the plane transverse to the beam line) with $|\eta| < 2.5$. Jets are concentrated showers of particles produced by a quark or gluon hadronizing in the detector. Jets were clustered using the default clustering algorithm with a cone size of $\Delta R = 0.4$, where $\Delta R = \sqrt{\eta^2 + \phi^2}$. We took the barrel-shaped region of CMS, $-1.5 \leq \eta \leq 1.5$ and $0 \leq \phi \leq 2\pi$, and filled 2-channel 96x90 grids by dropping the energy in each calorimeter tower into the nearest η, ϕ bin; the electromagnetic and hadronic calorimeters in CMS have different granularities. 96x90 images were extended to 96x96 as in the first generation. Example $t\bar{t}$ and $W + 4jets$ event calorimeter images from these second generation samples are shown as Figure 3 and Figure 4.

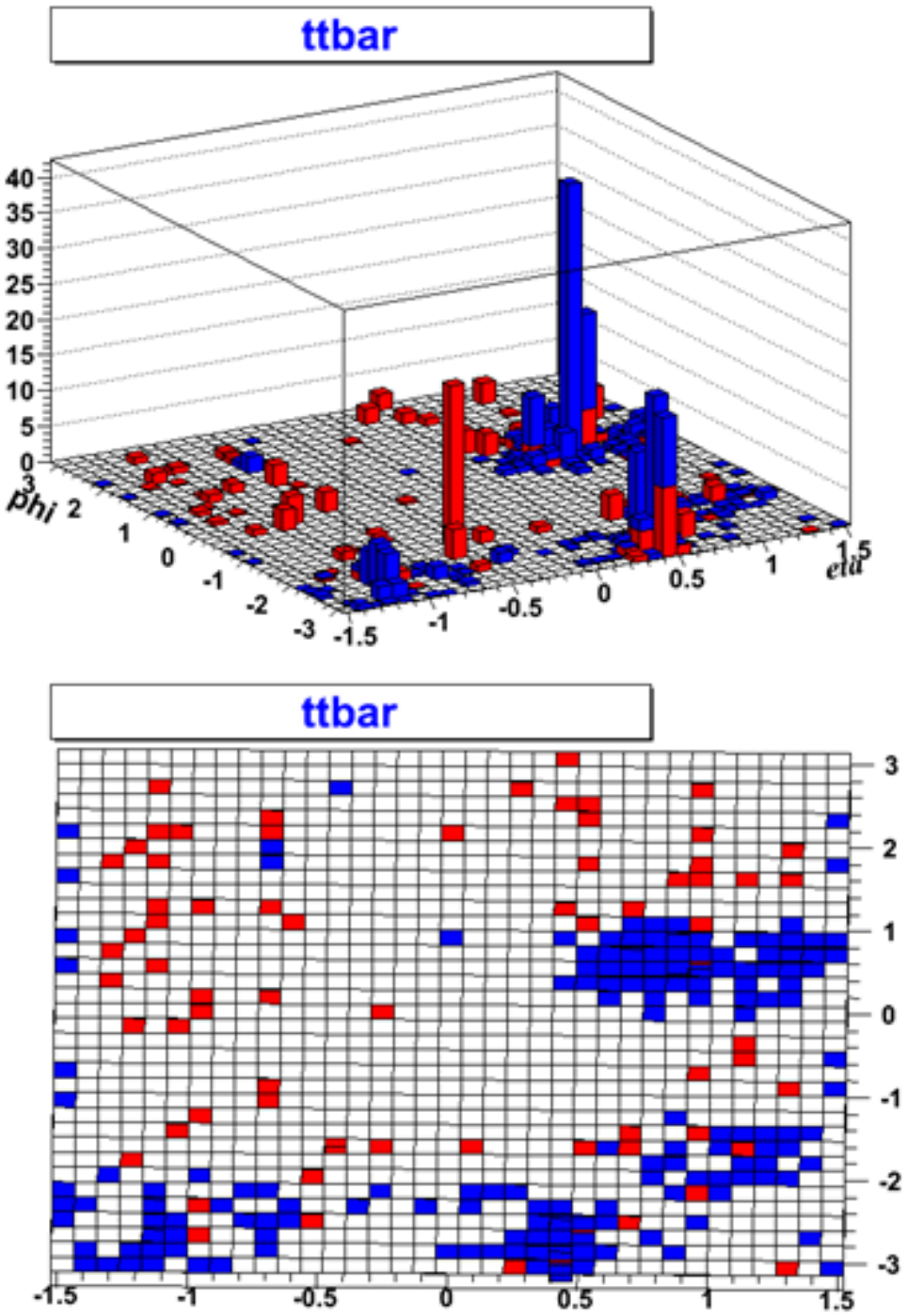

Figure 3: An example $t\bar{t}$ event calorimeter image.

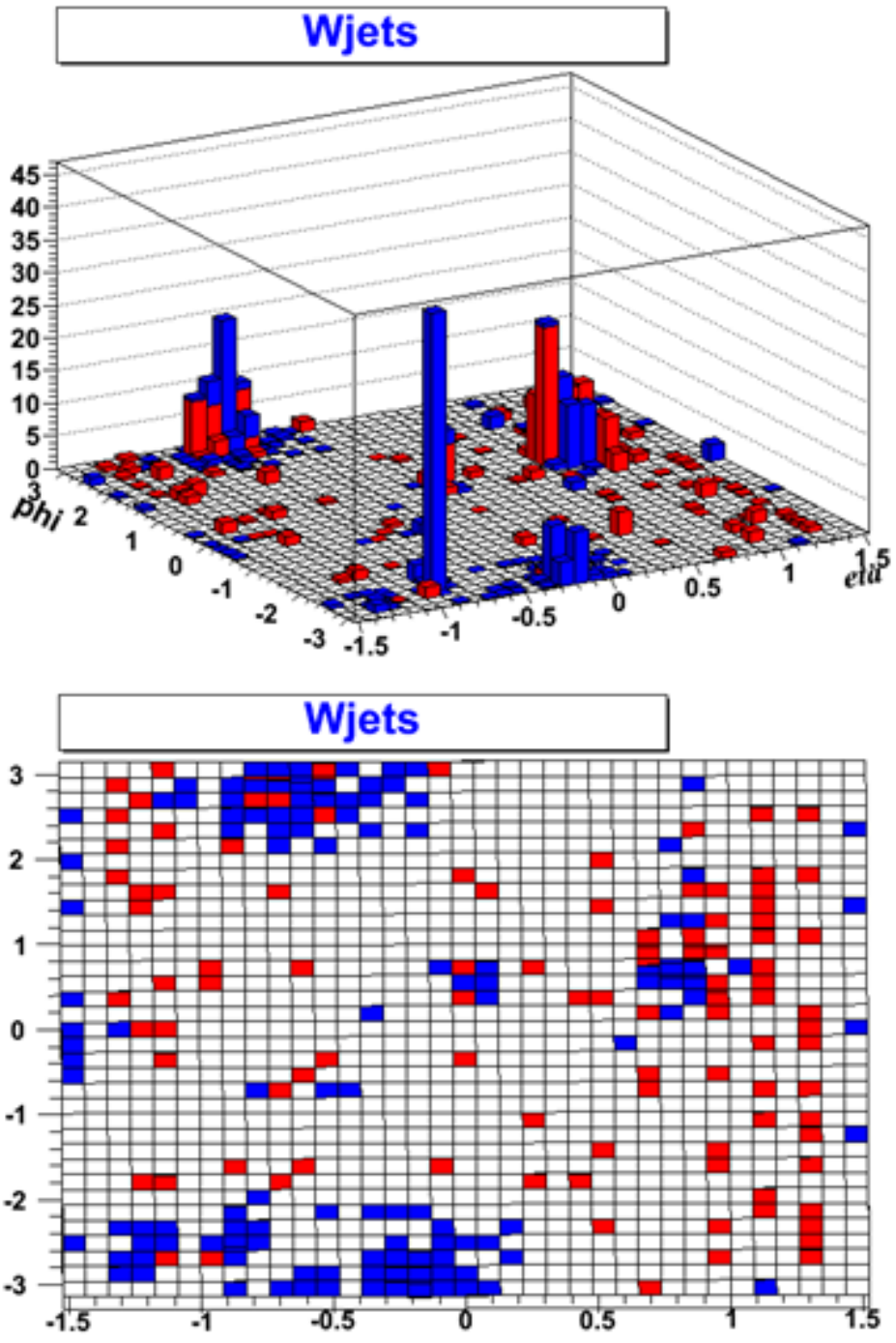

Figure 4: An example $W + 4jets$ event calorimeter image.

Chapter 4: Implementation and Development

Framework

We used Torch7, an open-source machine learning framework, to define and train our DCNNs (Collobert, Kavukcuoglu, & Farabet, 2011). Torch7 was chosen for its flexible model definition/training system, Graphics Processing Unit (GPU) acceleration capabilities, and cost (free). GPU acceleration was necessary to train DCNNs on reasonable time scales: a few days as opposed to weeks on conventional processors. We facilitated training on the GPU-accelerated nodes of the Ohio Supercomputer Center, which was free for approved academic projects such as ours.

Training Scheme

We began development with the first generation of $t\bar{t}$ and $W + 4jets$ samples. We partitioned the 80,000 $t\bar{t}$ events and 80,000 $W + 4jets$ events into three sets: the training set, the validation set, and the test set. The training set was used to adjust neuron connection weights via gradient descent. The validation set was run on periodically to track progress, which was available to induce higher-order training schemes if desired. The test set was held out as completely independent of the training scheme so that it could be used to indicate the true performance of the DCNN. The training set contained the vast majority of events, 78,000 $t\bar{t}$ events and 78,000 $W + 4jets$ events. The validation and test sets each consisted of 1,000 $t\bar{t}$ events and 1,000 $W + 4jets$ events.

The DCNN training scheme was as follows. First, shuffle the training set. Repeatedly propagate a batch of 32 events at a time through the DCNN and update neuron connection weights to reduce the error. Once the training set has been exhausted, reshuffle, and repeat. After every 20,000 training events are processed, classify events in the validation set with the DCNN (no weights are updated) and record the percent of correctly classified events. Induce higher-order training schemes with this information, such as a learning rate decay schedule. A batch size of 1 would correspond to stochastic gradient descent and generally produces best results, but a batch size of 32 is necessary to take advantage of GPU acceleration (Collobert, Kavukcuoglu, & Farabet, 2011). Finally, choose the iteration of the DCNN that performed the best on the validation set, and evaluate its performance on the independent test set. This represents the true performance of the DCNN's "best self," and the performance on the test set corresponds to the DCNN's performance expected when applied to new samples. Therefore, the performance on the test set can be used to make fair comparisons. DCNNs were allowed 3 days to train. Although a finite training time does not guarantee the attainment of asymptotic performance, performance gains appeared to become small near the end of this time period in general.

Base Model and Variations

There are many free parameters when defining a DCNN: the number of convolutional-subsampling layers, the number of fully connected layers, the number of features in each convolutional layer, the size of the features in each convolutional layer (3x3, 4x4, etc.), the subsampling size (2x2, 3x3, etc.), and the number of neurons in each

fully connected layer. The optimal DCNN must be found by empirically evaluating each choice of these parameters. We defined our base model DCNN through inspiration from the winning DCNN in the Galaxy Zoo challenge, a machine learning competition concerned with classifying galaxy morphologies (Dieleman, n.d.). This competition provided an example of a highly sophisticated application of DCNNs to non-natural images with features one might expect to be somewhat similar to our calorimeter image features. At the least, galaxy features appear more similar than the features associated with natural images (cats, dogs, and environments) of which most image classification competitions are concerned with.

Our base model was as follows:

1. input 2-channel calorimeter image
2. convolutional layer with 128 7x7 features
3. 3x3 subsampling layer
4. convolutional layer with 256 4x4 features
5. 3x3 subsampling layer
6. convolutional layer with 1024 4x4 features
7. 2x2 subsampling layer
8. 256 fully connected neurons
9. 256 fully connected neurons
10. 2 Softmax output neurons (estimated class membership probabilities).

The 7x7 features in the first layer corresponded to 0.44 in η and 0.49 in ϕ for the first generation samples and 0.22 in η and 0.49 in ϕ for the second generation samples:

comparable to the cone size used for jet reconstruction mentioned earlier. The pixel intensities in the input layer were normalized by subtracting the mean pixel intensity and dividing by the standard deviation of pixel intensity on a per-channel basis; normalizing input units generally produces best results in neural networks (LeCun, Bottou, Orr, & Müller, 2012). The weights in the network were randomly initialized in an appropriate fashion by Torch7. We chose to use rectified linear units as our activation functions (except for the Softmax final layer) because they have been shown to increase learning speed (Krizhevsky, Sutskever, & Hinton, 2012). We trained DCNNs similar to our base model, with one of the many free parameters varied, to search for the optimal DCNN. In addition, we tried several advanced training techniques: Dropout, data augmentation, and a learning rate decay schedule.

Dropout

Dropout is a technique that involves removing each neuron in a model with 50% probability during each training batch (Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 2012). During test time, all of the neurons are used, but with connective weights divided by 2. Dropout can be thought of as encouraging neurons to not depend highly on other each other or as a computationally efficient way to average a very large number of different models. Dropout has been shown to reduce overfitting, which occurs when a network has tailored itself to its training set at the expense of performing well on unseen data. We applied Dropout to the layers of fully connected neurons.

Data Augmentation

Another way to prevent overfitting is to increase the size of the training set (LeCun & Bengio, 1995). The detector and events are axially symmetric, so we introduced a random rotation in the ϕ direction before each training event is presented to the DCNN. The events are also symmetric about the plane transverse the beam line passing through the interaction point ($\eta = 0$), so we introduced a random flip in η as well. This effectively increased the size of the training set by a factor of 180 (90 for ϕ rotations and 2 for η flips) without the need to generate more events. Event generation on large experiments like CMS is a limited resource due to competition among many analyses, and data augmentation helps data-hungry methods like DCNNs cope with this reality.

Learning Rate Decay Schedule

The "early stopping learning rate decay schedule" is a technique that halves the DCNN learning rate, a parameter that controls how aggressively the network adjusts its connective weights to reduce error, when improved performance on the validation set is not seen 200,000 training events ahead. It facilitates this 200,000 event forecast by saving an earlier version of the network to revert back to. As discussed in Chapter 2, adjusting the weights in the DCNN can be thought of as a high-dimensional gradient descent problem where one is trying to find a minimum of a surface, the network's overall classification error. Taking steps in the direction opposite the gradient will generally lead to a minimum, but if the steps are too large, one will pass over and fail to converge onto

narrow minima. The “early stopping learning rate decay schedule” aims to identify this situation and reduces the step size (learning rate) to remedy it.

First Generation Studies

As an example of a particular DCNN, Figure 5 shows the base model’s performance on the validation set during training, with the usage of Dropout and data augmentation. It can be seen that the DCNN learns to distinguish $t\bar{t}$ and $W + 4jets$ as more training events are processed and gradient descent updates the neuron connection weights. The base model’s best performance on the validation set occurs near event 7 million, and its associated test set performance is 75.4%. To test the previously mentioned variations available, many DCNNs were trained with one aspect varied at a time. Generally, the training behavior appeared similar to Figure 5, but the results still provide valuable insight into the search for the optimal DCNN and training scheme.

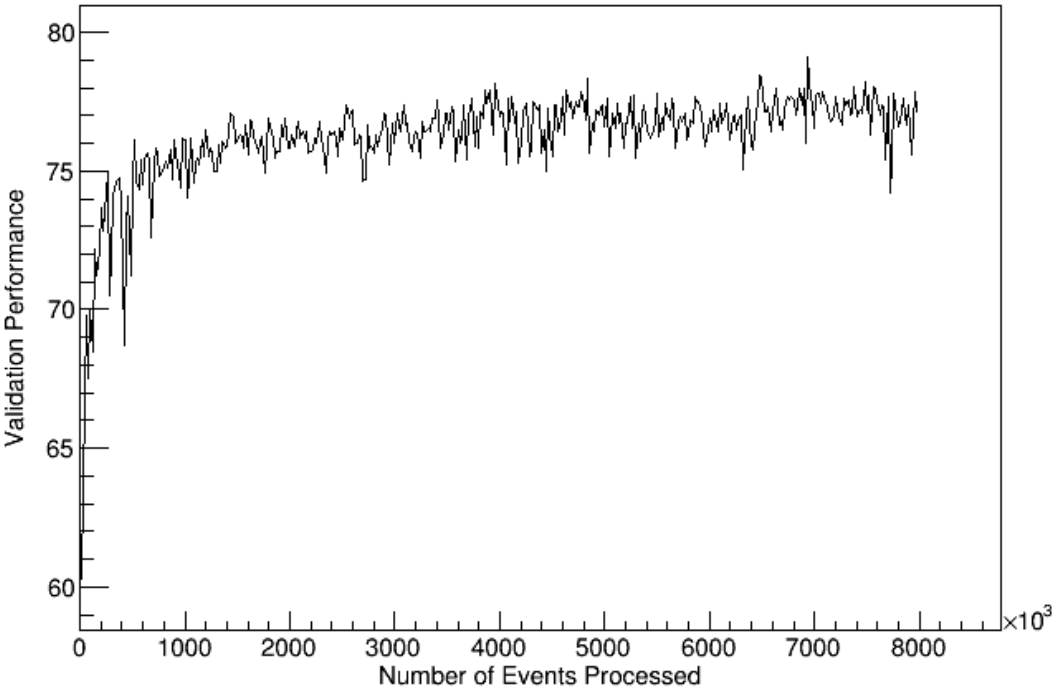

Figure 5: Example DCNN training progress.

Table 1 summaries the performances of a variety of DCNNs trained with Dropout, random ϕ rotation data augmentation, and the "early stopping learning rate decay schedule." The variations in performance are generally less than 1%, but one may attempt to interpret trends. One may expect an approximately ±0.3% fluctuation for different runs of the same DCNN with a different random number generator seed.

1. Adding more or having fewer fully connected layers does not seem to bolster performance.
2. Removing the last subsampling layer (3 convolutional layers and 2 subsampling layers) appears to improve performance; this is consistent with the intuition that the final subsampling layer is "throwing away" information.
3. Too many or too few feature maps in the convolutional layers appears to hinder performance; perhaps too few features are not sufficient to represent the calorimeter images, and too many features reduces the number of gradient descent steps that can be computed in the 3 day training period.
4. A small number of neurons in the fully connected layers appears to degrade performance; one would imagine that the function necessary to estimate the class membership probabilities is not simple.
5. The size of features in the convolutional layers appears to have an effect on performance; the direction of which may be a complex trade-off of computational cost and representational power.

Table 1: Variations on the DCNN base model.

Variation on Base Model	**Performance on Test Set (%)**
Number of Fully Connected Layers	
1 fully connected layer	73.75
2 fully connected layers (Base Model)	74.45
3 fully connected layers	74.00
Number of Convolutional/Subsampling Layers	
3 convolutional layers; 3 subsampling layers (Base Model)	74.45
3 convolutional layers; 2 subsampling layers	74.70
2 convolutional layers; 2 subsampling layers	74.30
2 convolutional layers; 1 subsampling layer	74.15
1 convolutional layer; 1 subsampling layer	73.10
Number of Features in Convolutional Layers	
16 features in conv. layer 1; 32 features in conv. layer 2; 64 features in conv. layer 3	74.00
32 features in conv. layer 1; 64 features in conv. layer 2; 128 features in conv. layer 3	74.75
128 features in conv. layer 1; 256 features in conv. layer 2; 1024 features in conv. layer 3 (Base Model)	74.45
256 features in conv. layer 1; 512 features in conv. layer 2; 1024 features in conv. layer 3	73.95
Number of Neurons in Fully Connected Layers	
32 neurons per fully connected layer	73.25
128 neurons per fully connected layer	73.55
256 neurons per fully connected layer (Base Model)	74.45
512 neurons per fully connected layer	74.45
1024 neurons per fully connected layer	74.70
4096 neurons per fully connected layer	74.45
Sizes of Convolutional Layer Features and Subsampling Layers	
convolutional layer feature sizes: 3x3, 4x4, 3x3; subsampling layer sizes: 2x2, 2x2, 2x2	74.90
convolutional layer feature sizes: 5x5, 5x5, 4x4; subsampling layer sizes: 2x2, 2x2, 2x2	74.85
convolutional layer feature sizes: 7x7, 4x4, 4x4; subsampling layer sizes: 3x3, 3x3, 2x2 (Base Model)	74.45
convolutional layer feature sizes: 9x9, 5x5, 5x5; subsampling layer sizes: 2x2, 2x2, 2x2	75.15
convolutional layer feature sizes: 13x13, 5x5, 4x4; subsampling layer sizes: 2x2, 2x2, 2x2	75.30
convolutional layer feature sizes: 15x15, 4x4, 4x4; subsampling layer sizes: 2x2, 2x2, 2x2	74.40

Table 2 shows the performance of the unaltered base model under the conditions of various special training procedures. Using Dropout instead of the “early stopping learning rate decay schedule” or a combination of the learning rate decay schedule and Dropout seems to be ideal. The addition of the data augmentation steps appears to improve performance in all cases.

Table 2: Variations on the DCNN training scheme.

Variation on Training Scheme				**Performance on Test Set (%)**
Random Phi Rotation	Random Eta Flip	Dropout	"Early Stopping Learning Rate Decay Schedule"	
✓	✓	✓		75.40
✓		✓		75.30
✓			✓	74.95
✓		✓	✓	74.45
		✓	✓	74.05

Scalable Dataset Management

Before tackling the second generation of samples, the data loading procedure in the DCNN training scheme was modified to accommodate large-scale datasets. Instead of loading entire datasets into memory, chunks of datasets are loaded as needed from a high-throughput Lustre parallel file system. This change was not necessary for the size of our CMS samples, but it may prove useful when we wish to train on larger datasets in the future. The new scheme made the batch-wise stochastic gradient descent slightly less random, as shuffling only occurs within chunks, but otherwise has no negative effects.

Second Generation Studies

The $W + 4jets$ CMS samples were generated with different ranges of generator-level HT, the scalar sum of the transverse momenta of generator-level particles: 100 GeV to 200 GeV, 200 GeV to 400 GeV, 400 GeV to 600 GeV, and 600 GeV to infinity. We determined that the range 200 GeV to 400 GeV constituted the vast majority of events passing our event selection criteria and therefore used that range for our $W + 4jets$ samples.

Based on the first generation sample studies, we chose to train the unaltered base model with all data augmentation steps and Dropout on the CMS $t\bar{t}$ and $W + 4jets$ samples. This served to evaluate the general effectiveness of DCNNs in a concrete setting. The training set contained the vast majority of events, 58,000 $t\bar{t}$ events and 58,000 $W + 4jets$ events. The validation and test sets each consisted of 1,000 $t\bar{t}$ events and 1,000 $W + 4jets$ events. Figure 6 shows the performance of the DCNN on the validation set as it trains; validation set performance was evaluated every 2,000 events. The performance on the test set associated with the highest validation set performance iteration was 71.8%. This value is not directly comparable to performance values obtained on the first generation samples (such as 75.4%), as the first generation samples correspond to an idealized detector.

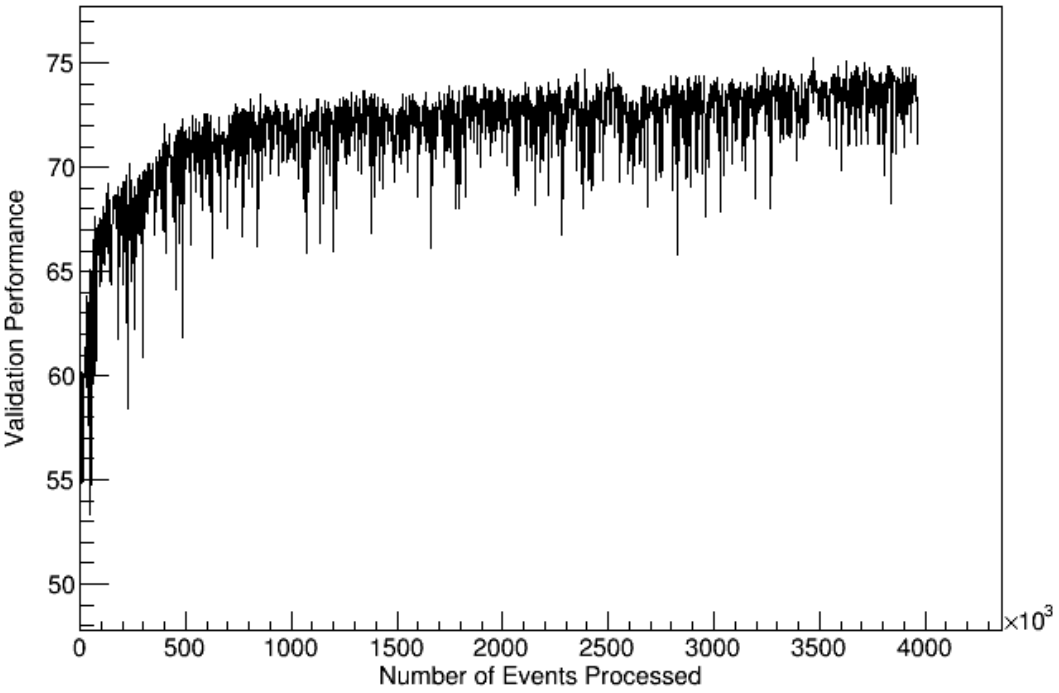

Figure 6: DCNN performance while training on the CMS samples.

Chapter 5: Performance Comparisons

The DCNN trained on the CMS samples achieved 71.8% correct classification on the independent test set. Other members of my group designed, trained, and tested an ordinary Artificial Neural Network (ANN) on the $t\bar{t}$ and $W + 4jets$ CMS samples. They partitioned the CMS samples into train, validation, and test sets in a similar fashion to the DCNN training scheme. Their ANN took as inputs the following engineered features, which were generated using information from all subdetector systems (not limited to the calorimeters as the DCNN was):

1 – 4. Et of the four highest Et jets

5. Sum of Et of jets with $|\eta| < 2.5$ and $Et > 25GeV$
6. Number of jets with $|\eta| < 2.5$ and $Et > 25GeV$
7. Number of jets with $|\eta| < 1.5$ and $Et > 25GeV$ (central jets)

where Et is energy in the plane transverse the beam line. The ANN had 3 fully connected layers of neurons: 7 input neurons in the first layer, 18 neurons in the second layer, and 1 output neuron in the third layer (the output was between 0 and 1, signal-like vs. background-like). The default activation functions, error function, and training scheme of JETNET were used (Peterson, Rögnvaldsson, & Lönnblad, 1994). The ANN achieved approximately 70% correct classification.

We estimated a 65% correct classification performance in a roughly equivalent 7 TeV $t\bar{t}$ analysis published by ATLAS by visual examination of figures (Aad et al., 2012). In a fully developed analysis such as this, simulated samples are generally degraded to match the data, which often results in a drop in performance. Our CMS samples are somewhat ideal in comparison, yet this provides another point of comparison.

Chapter 6: Future Work

The most immediate future work would be to apply DCNNs in a real, current analysis, such as the search for top quark produced in associated with the Higgs boson ($t\bar{t}H$) mentioned earlier. The estimated class membership probabilities (the DCNN outputs) could be conveniently used as extra input variables to existing multivariate analysis systems. If the DCNN is discovering features complementary to the features in current analyses, the sensitivity gains might be substantial. Systematically varying aspects to optimize a base model in the context of a full analysis would follow.

Further improvements to the DCNN scheme itself could be explored as well. Maxout functions have been shown to perform well with Dropout and could be used in place of the rectifying linear units (Goodfellow, Warde-Farley, Mirza, Courville, & Bengio, 2013). Since the events are symmetric in η, one might expect right-oriented features on the right side ($\eta > 0$) of the calorimeter images and left-oriented features on the left side ($\eta < 0$) of the calorimeter images. Perhaps the left side of the calorimeter images could be flipped, resulting in right-oriented features throughout the image. This technique may effectively reduce the number of features the DCNN needs to learn by a factor of 2, thus resulting in increased performance.

Adding More Subdetectors

The DCNNs discussed thus far only use information from the electromagnetic and hadronic calorimeters. ATLAS and CMS also have systems that record the trajectories of charged particles before they reach the calorimeters, as illustrated in green in Figure 7. Providing the DCNNs with this information will likely increase their performance. The most straightforward way to do so is to extrapolate charged particle trajectories onto the calorimeters and add two extra channels (or colors if you like) that contain the magnitude of the momentum of positively charged and negatively charged particles, respectively.

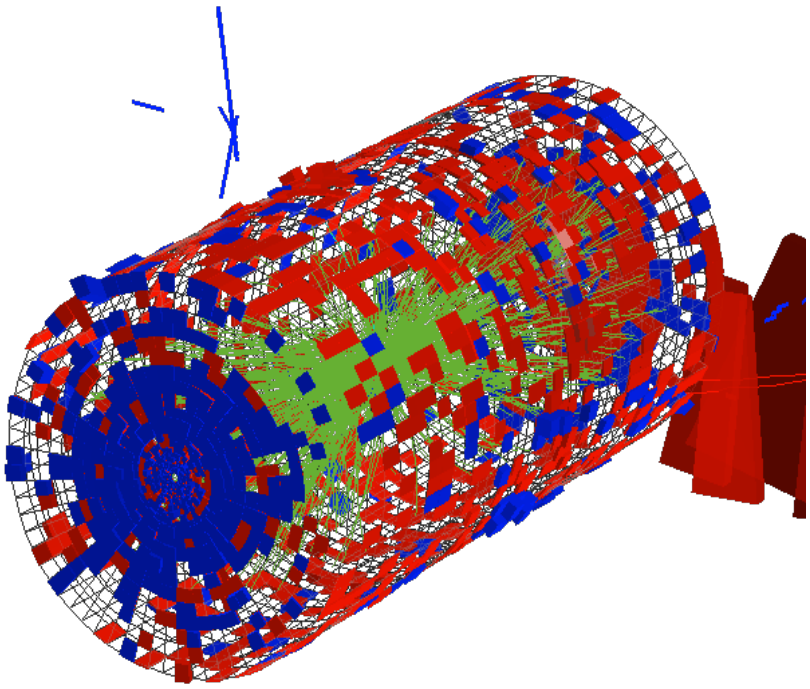

Figure 7: Energy deposits (red and blue) in cylindrical calorimeters and charged particle tracks (green) from inner tracking system.

Another way to use the charged particle trajectory information is to extrapolate the tracks inward toward the interaction point. If a heavy particle produced at the

interaction point travels a small distance and then decays, the tracks of its decay products will point to what is called a secondary vertex. Information from secondary vertices constitutes some of the most powerful engineered features (Chatrchyan et al., 2013). One crude way to allow a DCNN to "look" for secondary vertices would be to provide the DCNN with an image of a transverse slice of the detector near the interaction point with charged particle trajectories drawn, like in Figure 8 below.

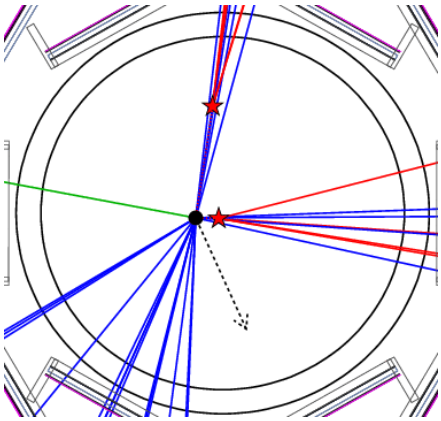

Figure 8: Transverse detector slice (beam line into the page) with secondary vertex (black dot) (Rappoccio, Costa, Sherman, Foland, & Franklin, n.d.).

Scene Labeling

Instead of simply classifying signal/background, DCNNs could be used to identify individual particles in the detector though a method called scene labeling. The objective of scene labeling is to label all pixels in an image with a class. Figure 9 shows a scene labeling algorithm identifying regions of sky, grass, etc. A DCNN trained in a special scale-invariant manner scans across the image, and a heuristic is used to aggregate classified pixels into smooth regions. The analogous operation in our detector would be to identify regions corresponding to different types of particles: *b* quarks, gluons,

photons, etc. This particle information could then be used as an input to a traditional feature engineering approach.

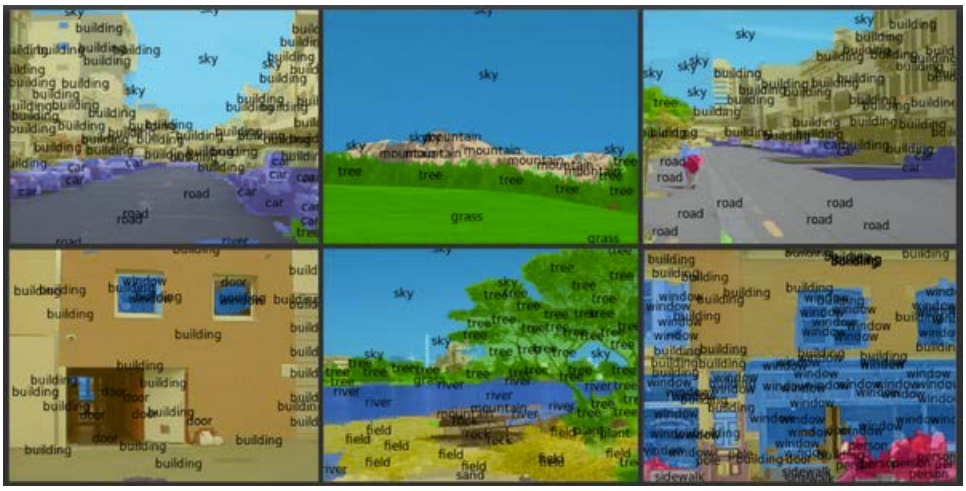

Figure 9: The output of a DCNN scene labeler on everyday images (Farabet, Couprie, Najman, & LeCun, 2013).

Online Application

DCNNs could also be applied in the data acquisition systems of experiments like CMS. The extremely high data rate in the CMS detector necessitates a system that quickly evaluates whether an event is interesting enough to be worth saving; this system is known as the trigger, and it is responsible for selecting the data samples on which all more thorough analyses are conducted. Currently, the trigger makes selections based on very simple, easy to compute criteria such as the clean reconstruction of a single, high momentum lepton. With the high energy and high luminosity environment at the LHC in 2015 and beyond, many more particles will clutter the detector and the efficiency of simple triggers will suffer.

The CMS inner tracking system is composed of silicon pixel and strip detectors that essentially read out a collection of 3D coordinates, called hits, due to the ionization left by charged particles passing through the detector. The 4 Tesla magnetic field due to CMS's superconducting solenoid causes charged particles travel in helical trajectories; the parameters of the helices allow the kinematic properties of the particles to be deduced. The combinatoric explosion associated with trying to fit a helix to every set of hits makes it difficult to include particle track information in the trigger. However, high momentum particle tracks are often indicative of rare, interesting phenomena and such information would certainly be valuable. A 3D DCNN, a generalization of a 2D DCNN in which convolutions are done in 3D (cubes instead of squares), could be trained to indicate the presence of high momentum tracks. A proposed upgrade to the CMS inner tracking system in 2023, shown in Figure 10 and Figure 11, could provide appropriate input. Since DCNNs are massively parallelizable (as evident by their implementation on GPUs), a DCNN pipeline could be configured to perform at trigger decision speeds with customized electronics.

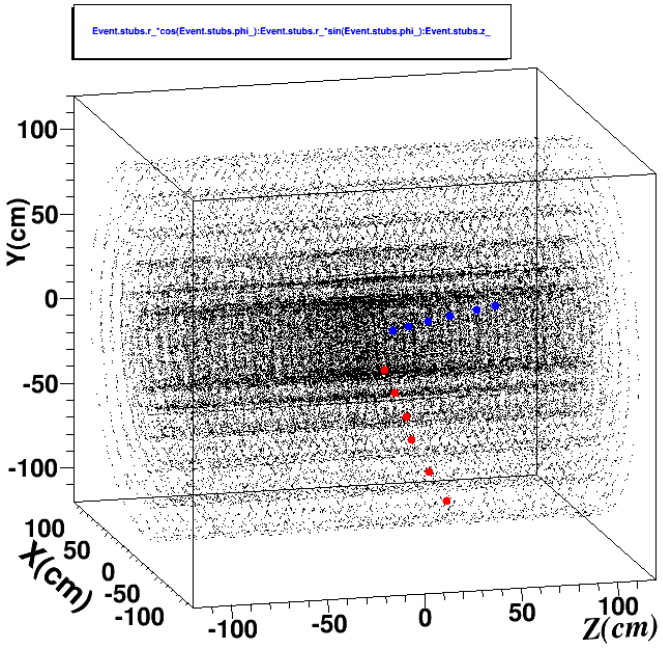

Figure 10: 3D view of proposed tracker upgrade; two particle tracks are highlighted in red and blue (B. Winer, personal communication, March 5, 2015).

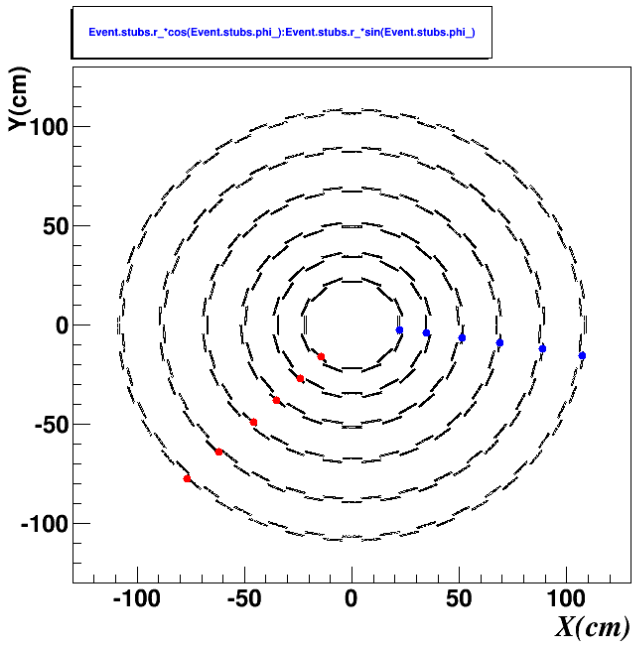

Figure 11: Transverse 2D view of proposed tracker upgrade (B. Winer, personal communication, March 5, 2015).

Chapter 7: Conclusion

We have demonstrated that a DCNN trained on only calorimeter data performs comparably to a traditional ANN when tasked to discriminating $t\bar{t}$ and $W + 4jets$ in the CMS detector under the conditions expected at the LHC in 2015 and beyond. This result demonstrates that DCNNs may indeed be able to help searches for rare, interesting phenomena gain critical sensitivity. The further investigation of DCNNs in the context of high energy physics is well-motivated.

References

Aad, G., Abbott, B., Abdallah, J., Abdelalim, A. A., Abdesselam, A., Abdinov, O., ... & Allwood-Spiers, S. E. (2012). Measurement of the top quark pair production cross-section with ATLAS in the single lepton channel. *Physics Letters B*, *711*(3), 244-263.

Alwall, J., Herquet, M., Maltoni, F., Mattelaer, O., & Stelzer, T. (2011). MadGraph 5: going beyond. *Journal of High Energy Physics*, *2011*(6), 1-40.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, *19*, 153.

Bernreuther, W. (2008). Top quark physics at the LHC. *Journal of Physics G: Nuclear and Particle Physics*, *35*(8).

Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing* (pp. 227-236). Springer Berlin Heidelberg.

Bruce, R., Arduini, G., Fartoukh, S., Giovannozzi, M., Lamont, M., Metral, E., ... & Wenninger, J. (2014). Baseline LHC machine parameters and configuration of the 2015 proton run. *arXiv preprint arXiv:1410.5990*.

Chatrchyan, S., Khachatryan, V., Sirunyan, A. M., Tumasyan, A., Adam, W., Aguilo, E., ... & Zuyeuski, R. (2012). Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Physics Letters B*, *716*(1), 30-61.

Chatrchyan, S., Khachatryan, V., Sirunyan, A. M., Tumasyan, A., Adam, W., Bergauer, T., ... & Van Spilbeeck, A. (2013). Search for the standard model Higgs boson produced in association with a top-quark pair in pp collisions at the LHC. *Journal of High Energy Physics*, *2013*(5), 1-47.

Collobert, R., Kavukcuoglu, K., & Farabet, C. (2011). Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop* (No. EPFL-CONF-192376).

de Favereau, J., Delaere, C., Demin, P., Giammanco, A., Lemaître, V., Mertens, A., & Selvaggi, M. (2013). DELPHES 3, A modular framework for fast simulation of a generic collider experiment. *arXiv preprint arXiv:1307.6346*.

Dieleman, S. (n.d.). My solution for the Galaxy Zoo challenge. Retrieved December 19, 2014, from http://benanne.github.io/2014/04/05/galaxy-zoo.html

Evans, L., & Bryant, P. (2008). LHC machine. *Journal of Instrumentation*, *3*(08), S08001.

Farabet, C., Couprie, C., Najman, L., & LeCun, Y. (2013). Learning hierarchical features for scene labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *35*(8), 1915-1929.

Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., & Bengio, Y. (2013). Maxout Networks. In *Proceedings of The 30th International Conference on Machine Learning* (pp. 1319-1327).

Higgs, P. W. (1964). Broken Symmetries and the Masses of Gauge Bosons. Physical Review Letters, 13(16), 508.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Index of /tutorial/_images. (2015, May 4). Retrieved May 5, 2015, from http://deeplearning.net/tutorial/_images/mylenet.png

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks, 3361*.

LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K. R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade* (pp. 9-48). Springer Berlin Heidelberg.

Orimoto, T. J. (2009). First CMS Results with LHC Beam. *arXiv preprint arXiv:0905.4814*.

Peterson, C., Rögnvaldsson, T., & Lönnblad, L. (1994). JETNET 3.0—A versatile artificial neural network package. *Computer Physics Communications,81*(1), 185-220.

Rappoccio, S., Costa, J., Sherman, D., Foland, A., & Franklin, M. (n.d.). Top pair cross section in lepton jets events with secondary vertex b-tags. Retrieved December 19, 2014, from http://www-cdf.fnal.gov/physics/new/top/2005/ljets/xs_svx/public.bkp.html

Sjöstrand, T., Mrenna, S., & Skands, P. (2008). A brief introduction to PYTHIA 8.1. *Computer Physics Communications, 178*(11), 852-867.